DSHGEM-LIB　通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

# APP インタフェース
# ライブラリ関数説明書

( C, C++, .Net-Vb,C# )

## VOL-3／１５

３．４．　Limit 変数リミット情報関連関数

３．５．　TR 状態変数トレース情報アクセスサービス関数

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30323-00

**[取り扱い注意]**

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、(株) データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

| 番号 | 改訂日付 | 項 目 | 概 略 |
|---|---|---|---|
| 1. | 2009.6 | 改訂版 | 以前の DSHGEM-LIB-07-3032x-00 を全面改訂<br>.Net VB2008, C#2008 対応関数の説明を追加した。 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

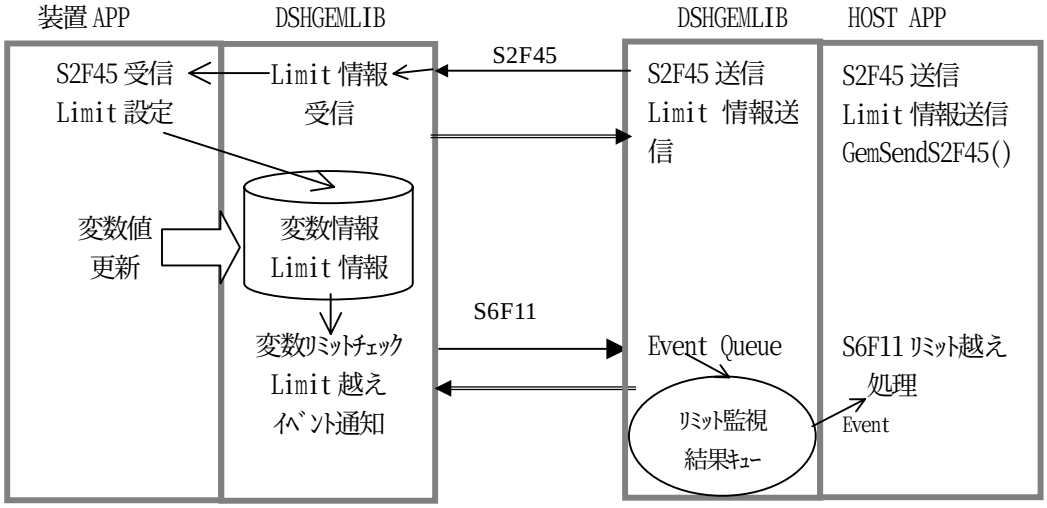
目　次

(VOL－4 に続く)

## ３．４．　Limit 変数リミット情報関連関数

変数については、３．３で述べましたが、ここでは変数リミットに関連する関数について説明します。
変数リミット情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスするために以下の DSHGEM-LIB ライブラリ関数を使用します。また、DSHGEM-LIB は変数値のリミット監視機能をおこなうための関数についても説明します。

変数リミット情報は、DSHGEM-LIB が管理しています。




（1）情報アクセスとリミット監視情報関連 API 関数

変数リミット情報アクセスに関連するサービスのための API 関数名は一覧表のとおりです。

| | API 関数名 | 機能 |
|---|---|---|
| 1 | GemSetMultiVLimit() | S2F45 ﾒｯｾｰｼﾞの内容で複数の変数ﾘﾐｯﾄ値を設定します。 |
| 2 | GemSetVLimit() | 1 個の変数ﾘﾐｯﾄ値を設定します。 |
| 3 | GemGetVLimit() | 変数ﾘﾐｯﾄ値を取得します。 |
| 4 | GemDelVLimit() | 変数ﾘﾐｯﾄ値を削除します。 |
| 5 | GemCheckVLimit() | 変数値のﾘﾐｯﾄﾁｪｯｸをします。 |
| 6 | GemSendS2F45() | 変数ﾘﾐｯﾄ属性定義 S2F45 ﾒｯｾｰｼﾞを送信します。 |
| 7 | GemSendS2F47() | 変数ﾘﾐｯﾄ属性要求 S2F47 ﾒｯｾｰｼﾞを送信します。 |
| 8 | GemSetupVLimitEventList() | 装置側 APP での変数ﾘﾐｯﾄｲﾍﾞﾝﾄ受信用ﾘｽﾄとｲﾍﾞﾝﾄの準備をします。 |
| 9 | GemTerminateVLimitEventList() | 装置側 APP での変数ﾘﾐｯﾄｲﾍﾞﾝﾄ受信用ﾘｽﾄとｲﾍﾞﾝﾄを終了します。 |
| 10 | GemPutVLimitEventList() | DSHGEM-LIB が APP での変数ﾘﾐｯﾄ監視ｲﾍﾞﾝﾄを通知します。 |
| 11 | GemGetVLimitEventList() | 装置側 APP が変数ﾘﾐｯﾄ監視ｲﾍﾞﾝﾄを受信用ﾘｽﾄから取得します。 |

リミット監視機能については、「**変数リミット監視機能　説明書**」を参照ください。

（2）ライブラリ関数

他に APP が使用できる変数リミット情報処理用 API 関数として、以下の関数があります。

| | API 関数名 | 機能 |
|---|---|---|
| 1 | DshEncodeS2F45() | TLIMIT_LIST 構造体の変数ﾘﾐｯﾄ属性情報を S2F45 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。 |
| 2 | DshDecodeS2F45( ) | S2F45 のﾒｯｾｰｼﾞ内変数ﾘﾐｯﾄ情報を TLIMIT_LIST 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。 |
| 3 | DshDecodeS2F46() | S2F46 のﾒｯｾｰｼﾞ内の変数ﾘﾐｯﾄ属性応答情報を TLIMIT_ERR_LIST 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。 |
| 4 | DshFreeTLIMIT_LIST() | 変数ﾘﾐｯﾄ情報が格納されている TLIMIT_LIST 構造体と内部で使用されている全ﾒﾓﾘを開放するための関数です。 |
| 5 | DshFreeTLIMIT_INFO() | 1 個の変数ﾘﾐｯﾄ情報が格納されている TLIMIT_INFO 構造体の内部で使用されている全ﾒﾓﾘを開放するための関数です。 |
| 6 | DshCopyTLIMIT_LIST() | TLIMIT_LIST の変数ﾘﾐｯﾄ情報を別の構造体にｺﾋﾟｰします。 |
| 7 | DshCopyTLIMIT_INFO() | TLIMIT_INFO、変数ﾘﾐｯﾄ情報構造体ﾒﾓﾘの内容を別の構造体にｺﾋﾟｰします。 |
| 8 | DshInitTLIMIT_LIST() | TLIMIT_LIST 変数ﾘﾐｯﾄ属性定義ﾘｽﾄ格納用構造体の初期設定を行います。<br>(TLIMIT_LIST は複数の TLIMIT_INFO を格納するための構造体) |
| 9 | DshAddTLIMIT_LIST() | TLIMIT_LIST 変数ﾘﾐｯﾄ属性定義ﾘｽﾄ格納用構造体内に TLIMIT_INFO 構造体に格納されている変数ﾘﾐｯﾄ情報を加えます。 |
| 10 | DshInitTLIMIT_INFO() | １個の変数のための TLIMIT_INFO 変数ﾘﾐｯﾄ属性定義ﾘｽﾄ格納用構造体内の初期設定を行います。 |
| 11 | DshPutTLIMIT_INFO() | TLIMIT_INFO 構造体に 1 個のﾘﾐｯﾄ ID の上下限ﾃﾞｰﾀを TLIMIT_INFO 構造体内に加えます。 |
| 12 | DshInitTLIMIT_ERR_LIST() | S2F45 の応答ﾒｯｾｰｼﾞ生成に使用する TLIMIT_ERR_LIST 応答情報構造体の初期設定を行います。このﾘｽﾄは TLIMIT_ERR_INFO 用格納ﾘｽﾄです。 |
| 13 | DshPutTLIMIT_ERR_LIST( | S2F45 の応答ﾒｯｾｰｼﾞ生成に使用する TLIMIT_ERR_LIST 応答情報構造体に 1 個の TLIST_ERR_INFO を設定をします。 |
| 14 | DshFreeTLIMIT_ERR_LIST() | 変数ﾘﾐｯﾄ情報が格納されている TLIMIT_LIST 構造体内部で使用されている全ﾒﾓﾘを開放するための関数です。 |
| 15 | DshInitTLIMIT_ERR_INFO() | S2F45 の応答ﾒｯｾｰｼﾞ生成に使用する TLIMIT_ERR_INFO 応答情報構造体の初期設定を行います。 |
| 16 | DshPutTLIMIT_ERR_ID() | S2F45 の応答ﾒｯｾｰｼﾞ生成に使用する TLIMIT_ERR_INFO 応答情報構造体にﾘﾐｯﾄ ID を設定します。 |
| 17 | DshFreeTLIMIT_ERR_INFO() | 1 個の変数ﾘﾐｯﾄ情報が格納されている TLIMIT_INFO 構造体の内部で使用されている全ﾒﾓﾘを開放するための関数です。 |
| 18 | DshMakeS2F45Response() | TLIMIT_LIST, TLIMIT_ERR_LIST から S2F46 ﾒｯｾｰｼﾞを生成します。 |
| 19 | DshEncodeVidList() | 変数 ID ﾘｽﾄに含まれている VID を DSHMSG 構造体の buffer にｴﾝｺｰﾄﾞします。<br>3.3.6.6 を参照ください。 |
| 20 | DshDecodeS2F48() | S2F48 ﾒｯｾｰｼﾞに含まれる変数ﾘﾐｯﾄ情報を TLIMIT_RSP_LIST 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。 |
| 21 | DshFreeTLIMIT_RSP_INFO() | S2F47 の応答ﾒｯｾｰｼﾞ S2F48 のﾃﾞｺｰﾄﾞ、格納した情報構造体内の内部で使用されている全ﾒﾓﾘを開放するための関数です。 |

| | | |
|---|---|---|
| 22 | DshFreeTVLIMIT_EVENT_INFO | TVLIMIT_EVENT_INFO 構造体内に使用されているﾒﾓﾘを開放します。 |

## ３．４．１　使用する情報格納構造体

変数リミット属性定義情報を操作する関数は、情報格納のための TLIMIT_LIST などの構造体を使用します。
TLIMT_LIST とのその内部で使用する他の構造体は下記のとおりです。

（1）**TLIMIT_LIST** - Limit Information List

```
typedef struct{
        int         vid_count;
        TLIMIT_INFO **limit_list;
} TLIMIT_LIST;
```

① vid_count = 0 の場合、変数ﾘﾐｯﾄ情報が１つもないことを意味します。
② vid_count > 0 の場合、limit_list[]内に変数ﾘﾐｯﾄ情報が格納されている TLIMT_INFO のﾎﾟｲﾝﾀが格納されています。

（2）**TLIMIT_INFO** - Limit Information

```
typedef struct{
        TDVID       vid;
        int         limit_count;
        TLIMIT_ID_INFO **limitid_list;
        int         format;             // for upperdb & lowerdb
        int         asize;              //   "         "   "
} TLIMIT_INFO;
```

① vid は変数ﾘﾐｯﾄを設定する変数 ID です。
② limit_count は、vid に対し何個のﾘﾐｯﾄ値があるかを示す個数です。
③ limitid_list[]には limit_count の数だけのﾘﾐｯﾄ ID と上下限値が格納されている TLIMIT_ID_INFO へのﾎﾟｲﾝﾀが格納されます
④ format は上下限値のﾃﾞｰﾀﾀｲﾌﾟです。
⑤ asize は上下限ﾃﾞｰﾀの配列ｻｲｽﾞです。

（3）**TLIMIT_ID_INFO** - Limit ID Information

```
typedef struct{
        TLIMITID    limit_id;
        void        *upperdb;               // 上限値
        void        *lowerdb;               // 下限値
} TLIMIT_ID_INFO;
```

① limit_id は vid に対するﾘﾐｯﾄ ID です。
② upperdb は上限値が格納されている領域のﾎﾟｲﾝﾀです。
③ lowerdb は下限値が格納されている領域のﾎﾟｲﾝﾀです。

（4）TLIMIT_ERR_LIST - S2F46 応答情報格納用構造体

```
typedef struct{
        int         vlaack;                 // S2F46 の VLAACK
        int         err_count;              // 含まれていたｴﾗｰ情報の数
        TLIMIT_ERR_INFO **limit_list;       // ｴﾗｰ情報のﾘｽﾄ
} TLIMIT_ERR_LIST;
```

（5）TLIMIT_ERR_INFO – S2F46 に含まれる１個のｴﾗｰ情報格納構造体

```
typedef struct{
        TDVID       vid;                    // 変数 ID
        int         lvack;                  // 変数 ID のﾘﾐｯﾄ情報に対する LVACK
        int         lmt_count;              // ﾘﾐｯﾄ ID の数(limt_id ﾘｽﾄ内の)
        TLIMITID    *limit_id;              // ﾘﾐｯﾄ ID 格納ﾘｽﾄ
        int         *limitack;              // limit_id 格納ﾘｽﾄ内のﾘﾐｯﾄ ID に対する ack
} TLIMIT_ERR_INFO;
```

（6）TLIMIT_RSP_LIST – S2F48 に含まれるﾘﾐｯﾄ情報ﾘｽﾄ格納構造体

```
typedef struct{
        int         vid_count;              //
        TLIMIT_RSP_INFO **limit_list;
} TLIMIT_RSP_LIST;
```

（7）TLIMIT_RSP_INFO – S2F48 応答情報内のﾘﾐｯﾄ情報格納構造体

```
typedef struct{
        TVID        vid;                    // 変数 ID
        char        *units;                 // 単位名
        int         format;                 // for upperdb & lowerdb
        int         asize;                  //   "         "   "
        void        *limit_min;             // 最小ﾘﾐｯﾄ値
        void        *limit_max;             // 最大ﾘﾐｯﾄ値
        int         limit_count;            // ﾘﾐｯﾄ ID の数
        TLIMIT_ID_INFO **limitid_list;      // ﾘﾐｯﾄ ID 情報ﾘｽﾄのﾎﾟｲﾝﾀ
} TLIMIT_RSP_INFO;
```

（8）TVLIMIT_EVENT_INFO – リミット監視結果情報格納構造体

```
typedef struct{
        TVID    vid;                    // vid
        char    *value;                 // value in ascii
        int     limitid;                // limit id
        int     dir;                    // transient direction (0=up, 1=down)
}TVLIMIT_EVENT_INFO;
```

## ３．４．２　Limit 変数リミット情報アクセスとリミット監視関連関数

### ３．４．２．１　GemSetMultiVLimit()　- 複数の変数データリミット値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetMultiVLimit(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TLIMIT_LIST *lmtlist            // LIMIT 情報ﾘｽﾄ構造体ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetMultiVLimit (
         ByVal eqid As Int32,
         ByRef lmtlist As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int GemSetMultiVLimit(
         int eqid,
         ref TLIMIT_LIST lmtlist );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

lmtlist

例えば S2F45 メッセージをエンコードするときに使用する LIMIT 情報リスト格納構造体のポインタです。

エンコードは、DshEncodeS2F45()関数を使用して行います。

APP が、自身で Limit 情報を lmtlist に設定することもできます。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | lmtlist の内容が正しくなかったので設定できなかった。 |

（4）説明

lmtlist のリミット情報を DSHGEM-LIB が管理する装置変数情報内に登録設定します。

もし、lmtlist 内のメンバーvid_count が=0 の場合は、DSHGEM-LIB 内に登録されている全 VID のリミット値を無効にします。

正常に設定できた場合は、関数の戻り値は 0 になります。

lmtlist に含まれるリミット情報の内容が１つでも正しくなかった場合、Limit 値の設定は行われません。

装置に S2F45 メッセージを送信する際に、S2F45 メッセージを DshEncodeS2F45()関数を使って TLIMIT_LIST 構造体を使用します。

また、APP 側で TLIMIT_LIST 構造体領域を設け、設定したいリミット情報をその中にセットした上で、本関数を使って DSHGEM-LIB に設定登録することができます。

（5）補足事項

lmtlist が指定する TLIMIT_LIST 構造体内には 設定したい vid 数と各 vid に対する以下のリミット情報が

格納されていなければなりません。
［TLIMIT_LIST 内の情報］

```
vid_counVID -1 --- limit_count1
                        limitid11, upperdb11, lowerdb11   (0 個以上のﾘｽﾄ)
                        limitid12, upperdb12, lowerdb12
                        .
                        .
                vid-2 --- limit_count2
                        limitid21, upperdb21, lowerdb21
                        .
                        .
```

vid_count =0 の場合は、全 VID に対するリミット値を無効にします。
各 vid について、limit_count の数が=0 の場合は、その vid についてはリミット値が無効となります。

DSHGEM-LIB との関連と全体の処理順序は次のようになります。矢印の方向に処理が行われます。

| APP プログラム | DSHENGLIB.DLL | | ホスト |
|---|---|---|---|
| APP は GemGetSecsMsgReq()関数で S2F45 を受け取る。 | ← 受信 MSG ｷｭｰに登録関数を呼出す、 | ← | S2F45 送信<br>ﾘﾐｯﾄ情報 |
| 以下取得した S2F45 を処理関数による処理を行う。<br>①DshDecodeS2F45()で Limit 情報をﾃﾞｺｰﾄﾞする。(lmtlist に)<br>②APP は lmtlist 内の Limit 情報を評価する。その結果を lmtlist 内の Ack 情報として設定する。<br>③評価結果が正常の場合、DshRegisterLimit()で → | Limit 情報を保存 | | |
| 戻り値が結果 ←<br>④DshResponseS2F46()で S2F46 ﾒｯｾｰｼﾞを応答する。 → | | → | S2F46 を受信する。 |

## ３．４．２．２　GemSetVLimit()　- 変数データリミット値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetVLimit(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        TLIMIT_INFO  *lminfo            // ﾘﾐｯﾄ情報格納構造体ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetVLimit (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef lminfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int GemSetVLimit(
        int eqid,
        uint vid,
        ref TLIMIT_INFO lminfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

vid

リミット値を設定する装置変数 ID です。

装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

lminfo

設定したいリミット値が格納されている構造体の格納ポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定された。 |
| (-1) | vid の値が正しくなかった。（登録されていなかった） |

（4）説明

vid で指定された装置変数に対し新しいリミット情報を設定します。
戻り値が設定結果になります。
正常に設定できた場合は、関数の戻り値は 0 になります。
指定された vid が登録されていない場合は(-1)が戻ります。

［TLIMIT_INFO 内の情報］

```
vid-1 --- limit_count
              limitid1, upperdb1, lowerdb1   (0 個以上のﾘｽﾄ)
              limitid2, upperdb2, lowerdb2
              .
```

limit_count = 0 の場合は、指定 vid に対し、リミット情報を無効にします。

## ３．４．２．３　GemGetVLimit()　- 変数データリミット値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVLimit(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        TLIMIT_INFO  *lminfo            // ﾘﾐｯﾄ情報格納構造体ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVLimit (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef lminfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int GemGetVLimit(
        int eqid,
        uint vid,
        ref TLIMIT_INFO lminfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

vid

リミット値を取得する装置変数 ID です。

装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

lminfo

取得したいリミット値が格納されている構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に取得できた。 |
| (-1) | vid の値が正しくなかった。（登録されていなかった） |

（4）説明

vid で指定された装置変数のリミット情報を DSHGEM-LIB から取得します。

戻り値が取得結果になります。

正常に取得できた場合は、関数の戻り値は 0 になります。

指定された vid が登録されていない場合は(-1)が戻ります。

[TLIMIT_INFO 内の情報]

```
vid-1 --- limit_count
              limitid1, upperdb1, lowerdb1   (0 個以上のﾘｽﾄ)
              limitid2, upperdb2, lowerdb2
              .
              .
```

limit_count = 0 の場合は、指定 vid に対し、リミット情報がないことを意味します。

## ３．４．２．４　GemDelVLimit()　- 変数データリミット値削除関数

（1）呼出書式

[c,C++]

```
API int APIX GemDelVLimit(
      int   eqid,                    // 通信対象装置 ID(0,1,2,...)
      TVID  vid                      // 変数 ID
);
```

[.NET VB]

```
Function GemDelVLimit (
       ByVal eqid As Int32,
       ByVal vid As Int32) As Int32
```

[.NET C#]

```
int GemDelVLimit(
       int eqid,
       uint vid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

vid

削除したいリミット値の装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に削除できた。 |
| (-1) | vid の値が正しくなかった。（登録されていなかった） |

（4）説明

vid で指定された装置変数のリミット情報を削除します。
戻り値が削除結果になります。
正常に削除できた場合は、関数の戻り値は 0 になります。
指定された vid が登録されていない場合は(-1)が戻ります。

## ３．４．２．５　GemCheckVLimit()　- 変数データ値のリミット値チェック関数

（1）呼出書式

[c,C++]

```
API int APIX GemCheckVLimit(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        TLIMITID  limitid,              // ﾘﾐｯﾄ ID
        void       *value               // ﾁｪｯｸ変数ﾃﾞｰﾀ値格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemCheckVLimit (
          ByVal eqid As Int32,
          ByVal vid As Int32,
          ByVal limitid As Int32,
          ByVal value As Int32) As Int32
```

[.NET C#]

```
int GemCheckVLimit(
          int eqid,
          uint vid,
          uint limitid,
          byte[] value );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

vid

チェックしたいリミット値の装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

limitid

指定変数データ情報内のチェックするリミットの limitid を指定する。

value

チェックしたい変数データ値が格納されている領域のポインタです。
データタイプは、vid によって決まりますので value の値もそれに合わせておかなければなりません。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | value がﾘﾐｯﾄ値内の範囲内であった。 |
| (-1) | vid の値が正しくなかった。（登録されていなかった） |
| (-2) | lowerdb 値よりも小さかった。 |
| (-3) | upperdb 値よりも大きかった。 |

（4）説明

装置変数の値 value を、vid の装置変数の limitid で指定されたリミット値(upperdb, lowerdb)の範囲内の値であるかどうかをチェックします。戻り値がチェック結果になります。
指定された vid が登録されていない場合は(-1)が戻ります。

## ３．４．２．６　GemSendS2F45() - 変数リミット属性リミットメッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F45(
        int  eqid,                      // 通信対象装置 ID(0,1,2,...)
        TLIMIT_LIST *list,              // 送信するﾘﾐｯﾄ定義情報ﾘｽﾄ格納構造体ﾎﾟｲﾝﾀ
        TLIMIT_ERR_LIST *errlist,       // S2F46 の応答情報格納用構造体ﾎﾟｲﾝﾀ
        int (WINAPI *callback)(),       // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                     // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F45 (
          ByVal eqid As Int32,
          ByRef info As dsh_info.TLIMIT_LIST,
          ByRef erlist As dsh_info.TLIMIT_ERR_LIST,
          ByVal callback As vcallback.callback_S2F45,
          ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F45(
          int eqid,
          ref TLIMIT_LIST info,
          ref TLIMIT_ERR_LIST erlist,
          CallbackS2F45 callback,
          uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

装置に送信したい変数リミット定義情報リストが格納されている構造体のポインタです。

errlist

装置から応答される S2F46 メッセージ内の情報をデコードした結果を格納するための構造体のポインタです。

変数値情報にはデータのフォーマット、配列サイズも与えられます。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。

ユーザは任意の関数名を指定できます。

このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | (1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。<br>errlist に受信した応答情報が格納されています。<br>(2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。 |
| (-1) | 送信できなかった。 |
| (-14) | T3 ﾀｲﾑｱｳﾄを検出した。(応答ﾒｯｾｰｼﾞが得られなかった) |

(4) 説明

装置に対し、list 内に指定された変数リミット定義情報を S2F45 メッセージを使って送信します。
正常に応答メッセージ S2F46 を受信できた場合は、そのメッセージをデコードし、errlist で指定された構造体リストに格納します。

また、引数 list で指定される TLIMIT_LIST 構造体内の vid_count が = 0 の場合は、装置に対し、全変数のリミット情報を未定義状態にすることを指示することになります。

送信要求から S2F45 応答メッセージ受信までの制御は引数の S2F45 コールバック関数指定の有無によって次のようになります。

| callback の指定 | 制御の流れ |
|---|---|
| なし<br>(=0) | S2F45 送信後、応答ﾒｯｾｰｼﾞ S2F46 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。<br>正常終了であれば、errlist に S2F46 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 |
| あり | 送信要求後、S2F45 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。<br>ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 errlist に S2F46 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。<br>ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。 |

正常に応答メッセージを受信した場合、errlist 構造体内に S2F45 に対する応答情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTLIMIT_ERR_LIST(errlist);
```

(5) コールバック関数

[c,C++]

```
API int APIX callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TLIMIT_ERR_LIST  *errlist,      // 応答情報が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S2F45(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erlist As
dsh_info.TLIMIT_ERR_LIST, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS2F45(int eqid, int end_status, ref TLIMIT_ERR_LIST erlist, uint upara);
```

end_status には以下の値が設定されます。

| 結果 | 意味 |
|---|---|
| 0 | 正常に送受信できた。 |
| (-1) | 送信ｴﾗｰを検出した。 |
| (-14) | T3 ﾀｲﾑｱｳﾄを検出した。 |

## ３．４．２．７　GemSendS2F47() - 変数リミット属性要求メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F47(
        int  eqid,                      // 通信対象装置 ID(0,1,2,...)
        TVID *vid_list,                 // 要求する VID のﾘｽﾄ領域のﾎﾟｲﾝﾀ
        int  count,                     // vid_list に含まれる VID の数
        TLIMIT_RSP_LIST *list,          // s2f48 で返された変数リミット属性情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int (WINAPI *callback)(),       // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                     // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F47 (
        ByVal eqid As Int32,
        ByRef vid_list As Int32,
        ByVal count As Int32,
        ByRef list As dsh_info.TLIMIT_RSP_LIST,
        ByVal callback As vcallback.callback_S2F47,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F47(
        int eqid,
        ref uint vid_list,
        int count,
        ref TLIMIT_RSP_LIST list,
        CallbackS2F47 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

vid_list

装置に要求したい変数リミット属性の変数 ID が格納されているリストのポインタです。

count

vid_list 内に含まれる変数リミット ID の数です。
count=0 の場合は、全変数が対象になります。

list

list 内に指定された変数 ID の変数名属性情報を格納するためのリストへのポインタです。
変数リミット属性情報には変数 ID、単位名、最大最小許容量、変数ﾘﾐｯﾄ ID とそれに対する上下限量，
が与えられます。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | (1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。<br>list に受信した変数リミット属性情報が格納されます。<br>(2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。 |
| (-1) | 送信できなかった。 |
| (-14) | T3ﾀｲﾑｱｳﾄを検出した。(応答ﾒｯｾｰｼﾞが得られなかった) |

（4）説明

装置に対し、vid_list 内に指定された count 分の変数 ID の変数リミット属性を S2F47 メッセージを使って要求します。

正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、list で指定された構造体リストに格納します。

また、count = 0 の場合は、リミット属性を持ち得る全変数が要求対象になります。

送信要求から S2F47 応答メッセージ受信までの制御は引数の S2F47 コールバック関数指定の有無によって次のようになります。

| callback の指定 | 制御の流れ |
|---|---|
| なし<br>(=0) | S2F47 送信後、応答ﾒｯｾｰｼﾞ S2F48 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。<br>正常終了であれば、list に S2F48 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 |
| あり | 送信要求後、S2F47 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。<br>ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 list に S2F48 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。<br>ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。 |

正常に応答メッセージを受信した場合、list 構造体内に変数リミット属性情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTLIMIT_RSP_LIST(list);
```

（5）コールバック関数

[c,C++]

```
API int APIX callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TLIMIT_RSP_LIST *list,          // リミット属性リストが格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S2F47(ByVal eqid As Integer, ByVal end_status As Integer, ByRef list As
dsh_info.TLIMIT_RSP_LIST, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS2F47(int eqid, int end_status, ref TLIMIT_RSP_LIST list, uint upara);
```

end_status には以下の値が設定されます。

| 結果 | 意味 |
|---|---|
| 0 | 正常に送受信できた。 |
| (-1) | 送信ｴﾗｰを検出した。 |
| (-14) | T3 ﾀｲﾑｱｳﾄを検出した。 |

## ３．４．２．８　GemSetupVLimitEventList() - 変数リミット監視結果リスト生成関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetupVLimitEventList(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        HANDLE event                    // 通知ｲﾍﾞﾝﾄ
);
```

[.NET VB]

```
Function GemSetupVLimitEventList (
          ByVal eqid As Int32,
          ByVal evt As Int32) As Int32
```

[.NET C#]

```
int GemSetupVLimitEventList(
          int eqid,
          uint evt );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

event

リミット監視において領域の遷移が起こった場合に DSHGEM-LIB から APP に通知するイベントのハンドルです。

CreateHandle()Windows 関数で生成したものを指定します。ただし、イベント通知を望まない場合は、NULL を指定してください。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定された。 |
| (-1) | 装置 ID が正しくなかった。 |

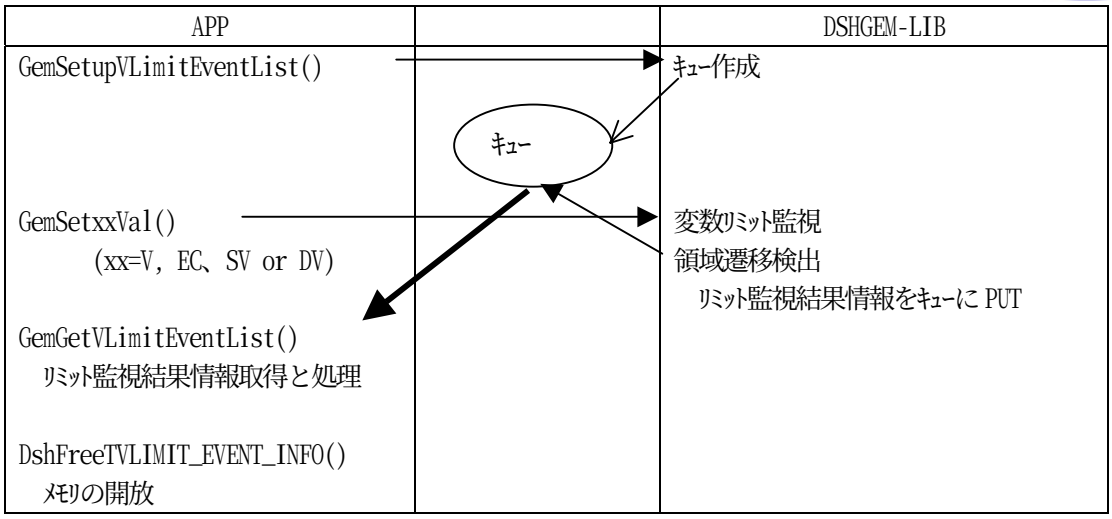
（4）説明

DSHGEM-LIB に対し、変数リミット監視によってデッドバンドを抜け、別の領域に遷移した場合、検出した対象の変数 ID、変数値、リミット ID ならびに遷移方向の情報（リミット監視結果情報）をキューと通して取得できるように専用キューを作成することを要求します。

event の値が有効（NULL でない）である場合は、領域遷移を検出した際、APP に対し event 通知するためイベントハンドルを指定します。

DSHGEM-LIB は最大３２個の情報を溜めることができるキューを作成します。

作成したキューからの監視結果情報の取得は、後述する GemGetVLimitEventList()関数を使って行います。監視結果情報が格納されている TVLIMIT_EVENT_INFO 構造体領域のポインタを得ることができます。


APP
DSHGEM-LIB
GemSetupVLimitEventList()
キュー作成
キュー
GemSetxxVal()
(xx=V, EC、SV or DV)
変数ﾘﾐｯﾄ監視
領域遷移検出
ﾘﾐｯﾄ監視結果情報をｷｭｰに PUT
GemGetVLimitEventList()
ﾘﾐｯﾄ監視結果情報取得と処理
DshFreeTVLIMIT_EVENT_INFO()
ﾒﾓﾘの開放

## ３．４．２．９　GemTerminateVLimitEventList () -変数リミット監視結果リスト終了関数

（1）呼出書式

[c,C++]

```
API void APIX GemTerminateVLimitEventList(
      int    eqid                        // 通信対象装置 ID(0,1,2,...)
);
```

[.NET VB]

```
sub GemTerminateVLimitEventList (
        ByVal eqid As Int32 )
```

[.NET C#]

```
void GemTerminateVLimitEventList(
        int eqid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

（3）戻り値

なし。

（4）説明

３．４．２．８の GemSetupVLimitEventList()関数で生成したキューの削除と指定されたイベントの開放を行うための関数です。

DSHGEM-LIB は、GemTerminateEquipment()関数が実行されたときに、本関数と同じ処理を行います。
従って、APP が明示的に行う必要がある場合にのみ使用してください。

## ３．４．２．10 GemPutVLimitEventList () - 変数リミット監視結果設定関数

（1）呼出書式

[c,C++]

```
GemPutVLimitEventList(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        VLIMIT_EVENT_INFO *info         // 監視結果格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemPutVLimitEventList (
          ByVal eqid As Int32,
          ByRef info As dsh_info.TVLIMIT_EVENT_INFO) As Int32
```

[.NET C#]

```
int GemPutVLimitEventList(
          int eqid,
          ref TVLIMIT_EVENT_INFO info );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

info

DSHGEM-LIBがリミット監視結果情報をリストに加えるリミット結果情報が格納されている構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | 装置IDが正しくない。 |
| (-2) | リストが満杯であったので設定できなかった。 |

（4）説明

infoで指定された構造体内のリミット監視結果情報をリストに加えます。
戻り値=0の場合、情報を設定できたことを意味します。

通常、本関数はDSHGEM-LIBが内部で使用します。

## 3．4．2．11 GemGetVLimitEventList () - 変数リミット監視結果取得関数

（1）呼出書式

[c,C++]

```
GemGetVLimitEventList(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        VLIMIT_EVENT_INFO *info         // 監視結果格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVLimitEventList (
          ByVal eqid As Int32,
          ByRef info As dsh_info.TVLIMIT_EVENT_INFO) As Int32
```

[.NET C#]

```
int GemGetVLimitEventList(
          int eqid,
          ref TVLIMIT_EVENT_INFO info );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

info

リミット監視結果情報をリストから取得し、その内容を格納するための構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | リストに情報が無かった。 |
| >0 | 取得できた。 |
| (-1) | 装置IDが正しくない。 |

（4）説明

APPが変数リミット監視結果情報がリストに残っているかどうかを調べ、残っていれば、1個分の情報をinfoで指定した構造体内に取得します。
戻り値が正の場合、情報を取得できたことを意味します。

取得したinfo情報の処理が終了した後は、info内に割当てられたメモリを開放するために次の関数を実行してください。

```
DshFreeTVLIMIT_EVENT_INFO( info );
```

TVLIMIT_EVENT_INFO内には、変数値がvalueメンバーで与えられますが、この値は、数値を文字列表現したものになります。

## ３．４．３　Limit 変数リミット関連ライブラリ関数

### ３．４．３．１　DshEncodeS2F45() - 変数リミット属性定義情報リストを S2F45 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS2F45(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S2F45 を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    bufflen,                 // buffer のﾊﾞｲﾄｻｲｽﾞ
        TLIMIT_LIST *pinfo              // ｴﾝｺｰﾄﾞしたい変数ﾘﾐｯﾄ情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS2F45 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef info As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int DshEncodeS2F45(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TLIMIT_LIST info );
```

（2）引数

smsg

エンコードした S2F45 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S2F45 のテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

エンコードしたい変数リミット属性定義情報リストが格納されている構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にｴﾝｺｰﾄﾞできた。 |
| (-1) | smsg を正しくｴﾝｺｰﾄﾞできなかった。 |

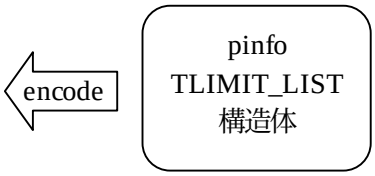
（4）説明

TLIMIT_LIST 情報リスト構造体に格納されている変数リミット情報を、S2F45 の SECS メッセージにエンコードします。

buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

smsg S2F45

```
L,m
  dataid
  L,2
    VID
      L,n
        L2,
          LIMITID,
          L p
            UPPERDB
            LOWERDB
        L,2
```



## ３．４．３．２　DshDecodeS2F45( ) - S2F45 を変数リミット情報リストにデコード

（1）呼出書式

**[c,C++]**

```
API int APIX DshDecodeS2F45(
        DSHMSG *smsg,                    // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TLIMIT_LIST *pinfo               // ﾃﾞｺｰﾄﾞした情報ﾘｽﾄを格納する構造体のﾎﾟｲﾝﾀ
);
```

**[.NET VB]**

```
Function DshDecodeS2F45 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef info As dsh_info.TLIMIT_LIST) As Int32
```

**[.NET C#]**

```
int DshDecodeS2F45(
          ref DSHMSG smsg,
          ref TLIMIT_LIST info );
```

（2）引数

smsg

S2F45 の SECS ﾒｯｾｰｼﾞ情報構造体のポインタです。

pinfo

デコードした結果の変数リミット情報を格納する構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にﾃﾞｺｰﾄﾞできた。 |
| (-1) | smsg を正しくﾃﾞｺｰﾄﾞできなかった。 |

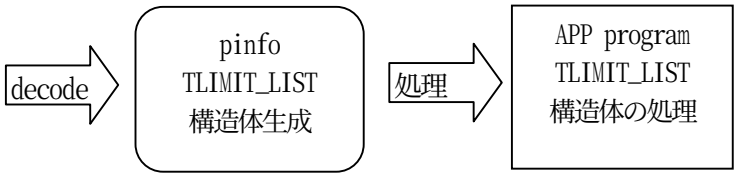
（4）説明

S2F45 メッセージに含まれる変数リミット情報リストを、ユーザプログラムが処理しやすい TLIMIT_LIST 構造体の中にデコードします。

変数リミット構造体メンバーのために使用するメモリは全て DSHGemLIB 側が準備します。

従って、**情報処理終了後は、pinfo のメモリを DshFreeTLIMIT_LIST( )関数を使って解放してください。**

```
smsg S2F45
  L,m
    dataid
    L,2
      VID
      L,n
          L2,
            LIMITID,
            L p
              UPPERDB
              LOWERDB
          L,2
```



（5）関連関数

DshEncodeLimitMSG( ) ： TLIMIT_LIST の内容を S2F45 メッセージにエンコードします。
DshFreeTLIMIT_LIST( ) ： TLIMIT_LIST のメモリを解放します。
DshCopyTLIMIT_LIST( ) ： TLIMIT_LIST の内容を別の構造体にコピーします。
DshMakeS2F45Response( ): TLIMIT_LIST の ACK 情報から S2F46 応答メッセージを作成します。

## ３．４．３．３　DshDecodeS2F46() - S2F46 を変数リミット応答情報リストにデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS2F46(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TLIMIT_ERR_LIST *pinfo          // ﾃﾞｺｰﾄﾞした応答情報ﾘｽﾄを格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS2F48 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef list As dsh_info.TLIMIT_RSP_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS2F46(
          ref DSHMSG smsg,
          ref TLIMIT_ERR_LIST erlist);
```

（2）引数

smsg

S2F46 の SECS ﾒｯｾｰｼﾞ情報構造体のポインタです。

pinfo

デコードした結果の変数リミット応答情報を格納する構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にﾃﾞｺｰﾄﾞできた。 |
| (-1) | smsg を正しくﾃﾞｺｰﾄﾞできなかった。 |

（4）説明

S2F46 メッセージに含まれる変数リミット応答情報リストを、ユーザプログラムが処理しやすい TLIMIT_ERR_LIST 構造体の中にデコードします。TLIMIT_ERR_LIST 内に複数の TLIMIT_ERR_INFO 構造体の情報が含まれます。

変数リミット構造体メンバーのために使用するメモリは全て DSHGEM-LIB 側が準備します。

従って、情報処理終了後は、pinfo のメモリを DshFreeTLIMIT_ERR_LIST()関数を使って解放してください。

```
smsg S2F46
  L,2
    vlaack
    L,m
      L,3
        VID
        LVACK
        L,n
            LIMITID,
            limitack
            .
```

## ３．４．３．４　DshFreeTLIMIT_LIST() - 変数リミット情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_LIST(
      TLIMIT_LIST *pinfo                    // ﾒﾓﾘを開放したい変数ﾘﾐｯﾄ情報ﾘｽﾄ構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_LIST (
        ByRef info As dsh_info.TLIMIT_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_LIST(
        ref TLIMIT_LIST info );
```

（2）引数

pinfo

メモリを解放したい変数リミット情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。

## ３．４．３．５　DshFreeTLIMIT_INFO() - 変数リミット情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_INFO(
      TLIMIT_INFO *pinfo                    // ﾒﾓﾘを開放したい変数ﾘﾐｯﾄ情報ﾘｽﾄ構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_INFO (
        ByRef info As dsh_info.TLIMIT_INFO)
```

[.NET C#]

```
void DshFreeTLIMIT_INFO(
        ref TLIMIT_INFO info );
```

（2）引数

pinfo

メモリを解放したい変数リミット情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。

## ３．４．３．６　LIMIT_LIST() - 変数リミット情報リスト構造体メモリのコピー

（1）呼出書式

[c,C++]

```
int     DshCopyTLIMIT_LIST(
        TLIMIT_LIST *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TLIMIT_LIST *sinfo                  // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTLIMIT_LIST (
        ByRef pinfo As dsh_info.TLIMIT_LIST,
        ByRef sinfo As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int DshCopyTLIMIT_LIST(
        ref TLIMIT_LIST pinfo,
        ref TLIMIT_LIST sinfo );
```

（2）引数

dinfo

コピー先の変数リミット情報リスト構造体のポインタです。

sinfo

コピー元の変数リミット情報リストが格納されている構造体メモリのポインタです。

（3）戻り値

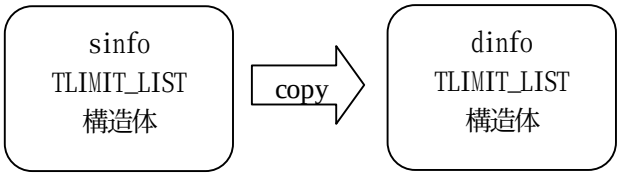
| 戻り値 | 意味 |
|---|---|
| 0 | 正常にｺﾋﾟｰできた。 |
| (-1) | sinfo または dinfo の値が NULL だったのでｺﾋﾟｰできなかった。 |

（4）説明

sinfo が指す TLIMIT_LIST 構造体内に格納されている変数リミット情報リストの内容を dinfo で指定されたメモリの TLIMIT_LIST 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用後、DshFreeTLIMIT_LIST()関数を使って開放してください。



## ３．４．３．７　DshCopyTLIMIT_INFO() - 変数リミット情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
int     DshCopyTLIMIT_INFO(
        TLIMIT_INFO *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TLIMIT_INFO *sinfo                  // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTLIMIT_INFO (
          ByRef pinfo As dsh_info.TLIMIT_INFO,
          ByRef sinfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int DshCopyTLIMIT_INFO(
          ref TLIMIT_INFO pinfo,
          ref TLIMIT_INFO sinfo );
```

（2）引数

dinfo

変数リミット情報構造体のコピー先構造体メモリのポインタです。

sinfo

コピー元の変数リミット情報が格納されている構造体メモリのポインタです。

（3）戻り値

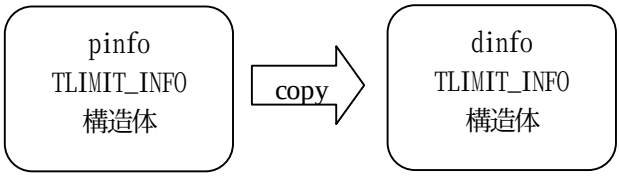
| 戻り値 | 意味 |
|---|---|
| 0 | 正常にｺﾋﾟｰできた。 |
| (-1) | sinfo またはdinfo の値が NULL だったのでｺﾋﾟｰできなかった。 |

（4）説明

sinfo が指すTLIMIT_INFO構造体内に格納されている変数リミット情報の内容をdinfo で指定された構造体メモリ内にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用後、DshFreeTLIMIT_INFO()関数を使って開放してください。



## ３．４．３．８　DshInitTLIMIT_LIST - 変数リミット属性定義情報 TLIMIT_LIST の初期設定

（1）呼出書式

[c,C++]

```
API void  APIX DshInitTLIMIT_LIST(
      TLIMIT_LIST *list,                  // TLIMIT_LIST 構造体のﾎﾟｲﾝﾀ
      int          count                  // ﾘﾐｯﾄ属性情報を設定したいﾘ変数の数
);
```

[.NET VB]

```
Sub DshInitTLIMIT_LIST (
        ByRef list As dsh_info.TLIMIT_LIST,
        ByVal count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_LIST(
        ref TLIMIT_LIST list,
        int count );
```

（2）引数

list

変数リミット属性定義情報格納用 TLIMIT_LIST 構造体のポインタです。このメンバーを初期設定します。

count

list に設定したい変数の数です。
count の値が =0 の場合は、全変数のリミット属性定義が未定義にすることになります。

（3）戻り値

なし。

（4）説明

本関数は S2F45 変数リミット属性定義メッセージを送信する際に使用することができます。
最初に list 内をクリアします。そして、引数で指定された count 分の変数のリミット属性定義に必要な TLIMIT_INFO 構造体のポインタリストを準備します。count は vid_count メンバーに設定されます。

1 個の変数のリミット属性定義情報を list 内に加えるために DshAddTLIMIT_INFO()関数が準備されています。

TLIMIT_LIST 構造体の使用後は DshFreeTLIMIT_LIST()関数を使って構造体内部で使用したメモリを開放してください。

## ３．４．３．９　DshAddTLIMIT_LIST() -変数リミット属性定義情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshAddTLIMIT_LIST(
        TLIMIT_LIST *list,              // TLIMIT_LIST 構造体のﾎﾟｲﾝﾀ
        TLIMIT_INFO *info               // 追加するﾘﾐｯﾄ属性定義情報構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshAddTLIMIT_LIST (
          ByRef list As dsh_info.TLIMIT_LIST,
          ByRef info As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int DshAddTLIMIT_LIST(
          ref TLIMIT_LIST list,
          ref TLIMIT_INFO info );
```

（2）引数

list

変数属性定義データを加えるの変数リミットリスト情報構造体のポインタです。

info

１個の変数に対するリミット属性定義情報がが格納されている構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に追加できた。 |
| (-1) | list が既に満杯になっていた。 |

（4）説明

先にDshInitTLIMIT_LIST()で初期設定された変数リミット属性定義リストにinfoで指定された属性定義を１個加えます。

info 内にはリミット ID,上下限データ値が格納されていなければなりません。

正常に加えることができたら 0 を返却します。
もし、list 内のvid_count メンバーで指定された分の変数リミット属性定義が既に設定されていた場合は、(-1)を返却することによって既に満杯になっていたことを知らせます。

## ３．４．３．10　DshInitTLIMIT_INFO - 変数リミット属性定義情報 TLIMIT_INFO の初期設定

（1）呼出書式

[c,C++]

```
API void  APIX DshInitTLIMIT_INFO(
        TLIMIT_INFO *info,              // TLIMIT_INFO 構造体のﾎﾟｲﾝﾀ
        int         vid,                // 変数 ID
        int         fmt,                // 変数のﾌｫｰﾏｯﾄ
        int         asize,              // 変数ﾃﾞｰﾀの配列ｻｲｽﾞ
        int         count               // 設定したいﾘﾐｯﾄ ID(ﾃﾞｰﾀ)数
);
```

[.NET VB]

```
Sub DshInitTLIMIT_INFO (
        ByRef info As dsh_info.TLIMIT_INFO,
        ByVal vid As Int32,
        ByVal fmt As Int32,
        ByVal asize As Int32,
        ByVal count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_INFO(
        ref TLIMIT_INFO info,
        uint vid,
        int fmt,
        int asize,
        int count );
```

（2）引数

info

変数リミット属性定義情報格納用 TLIMIT_INFO 構造体のポインタです。このメンバーを初期設定します。

vid

リミット情報を設定する対象の変数 ID です。

fmt

変数リミットデータのフォーマットです。(ICODE_A, ICODE_U1 などで指定します。)

asize

変数リミットデータの配列サイズです。

count

info に設定するリミット ID の数です。１個のリミット ID に対し UPPERDB, LOWERDB のリミット値が定義されます。

（3）戻り値

なし。

（4）説明

本関数は S2F45 変数リミット属性定義メッセージを送信する際に使用することができます。
最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。

info 内に count で指定された分の変数リミットデータ格納用 TLIMIT_ID_INFO 構造体ポインタリストを設けます。TLIMIT_INFO 内には 1 個の変数のためのリミット属性定義情報を格納することができます。

リミット ID と上下限リミットデータ行を info 内に順番に設定する関数として DshPutTLIMIT_INFO()が準備されています。

S2F45 送信については複数の変数のリミット属性定義を行うことができるため TLIMIT_INFO 構造体の集合を格納する TLIMIT_LIST 構造体の情報を使用します。

TLIMIT_INFO 構造体の使用後は DshFreeTLIMIT_INFO()関数を使って構造体内部で使用したメモリを開放してください。

## ３．４．３．11　DshPutTLIMIT_INFO() -変数リミット属性定義データの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTLIMIT_INFO(
        TLIMIT_INFO *info,              // TLIMIT_INFO 構造体のﾎﾟｲﾝﾀ
        int     limit_id,               // 変数ﾘﾐｯﾄ ID
        void    *upperdb,               // 上限ﾘﾐｯﾄﾃﾞｰﾀの格納ﾎﾟｲﾝﾀ
        void    *lowerdb                // 下限ﾘﾐｯﾄﾃﾞｰﾀの格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutTLIMIT_INFO (
          ByRef info As dsh_info.TLIMIT_INFO,
          ByVal limit_id As Int32,
          ByVal upperdb As IntPtr,
          ByVal lowerdb As IntPtr) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_INFO(
          ref TLIMIT_INFO info,
          int limit_id,
          byte[] upperdb,
          byte[] lowerdb );
```

（2）引数

info

変数属性定義データを加えるの変数リミット情報構造体のポインタです。

limit_id

リミット ID を指定します。１個の limit_id に対し上下限データが定義されます。

upperdb

リミットの上限値が格納されているメモリのポインタです。

lowerdb

リミットの下限値が格納されているメモリのポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に追加できた。 |
| (-1) | info 内が既に満杯になっていた。 |

（4）説明

先に DshInitTLIMIT_INFO()で初期設定された変数リミット情報構造体内の属性定義データリストに引数 limit_id, upperdb, lowerdb で指定された属性定義データを１個加えます。

upperdb, lowerdb には DshInitTLIMIT_INFO()関数で指定したフォーマットと配列サイズのデータを設定しなければなりません。

info 内の属性定義データリスト上に、本関数が実行される順に引数で与えられた属性定義データを追加していきます。

設定後 0 を返却します。

もし、info 内の limit_count メンバーで指定された分の属性定義データが既に設定済みであった場合は、(-1)を返却することによって既に満杯になっていたことを知らせます。

## ３．４．３．12　DshInitTLIMIT_ERR_LIST () - 変数リミット応答情報リストの初期化

（1）呼出書式

[c,C++]

```
API void APIX   DshInitTLIMIT_ERR_LIST(
        TLIMIT_ERR_LIST *list,              // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int         vlaack,                 // ackﾃﾞｰﾀ
        int         err_count               // ｴﾗｰ情報のﾘｽﾄｻｲｽﾞ（個数　0,1,2...）
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_LIST (
        ByRef list As dsh_info.TLIMIT_ERR_LIST,
        ByRef errinfo As dsh_info.TLIMIT_ERR_INFO) As Int32
```

[.NET C#]

```
void DshInitTLIMIT_ERR_LIST(
        ref TLIMIT_ERR_LIST list,
        int vlaack,
        int err_count );
```

（2）引数

list

TLIMIT_ERR_LIST 応答情報構造体のポインタです。

vlaack

vlaack - ACK の値です。

err_count

変数リミットエラー情報構造体の数です。 = 0 の場合はエラー情報がないことになります。

（3）戻り値

なし。

（4）説明

本関数は、S2F45 に対する応答メッセージのための応答情報を TLIMIT_ERR_LIST 構造体に初期設定します。
list で指定された構造体の vlaack メンバーに引数 vlaack の値を設定し、err_count メンバーに引数 err_count の値を設定します。
そして、もし、err_count > 0 の場合は、err_list に err_count だけの TERR_LIST エラー情報構造体のポインタリストを設けます。

err_info へのリミットエラー情報の設定には DshPutTLIMIT_ERR_LIST()関数を使用します。

## ３．４．３．13 DshPutTLIMIT_ERR_LIST () - 変数リミット応答情報の設定

（1）呼出書式

[c,C++]

```
API void APIX   DshPutTLIMIT_ERR_LIST (
        TLIMIT_ERR_LIST  *list,          // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TLIMIT_ERR_INFO  *info           // １個のｴﾗｰ情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_LIST (
          ByRef list As dsh_info.TLIMIT_ERR_LIST,
          ByRef errinfo As dsh_info.TLIMIT_ERR_INFO) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_ERR_LIST(
          ref TLIMIT_ERR_LIST list,
          ref TLIMIT_ERR_INFO errinfo );
```

（2）引数

list

プロセスジョブエラー情報構造体のポインタです。

info

１個のｴﾗｰ情報 TLIMIT_ERR_INFO のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | ﾘｽﾄが満杯で設定できなかった。 |

（4）説明

本関数は、list 内の limit_list リストの先頭から空きリストを探します。
もし、空きリストがなければ、(-1)を返却します。
もし、空きリストがあれば、その空きリストに info（ポインタ）を格納し、0 を返却します。本関数の実行前に DshInitTLIMIT_ERR_LIST()関数を使って list を初期化しておく必要があります。

info 内のメモリはそのまま list 内に使用されますので、info 内のメモリは、DshFreeTLIMIT_ERR_LIST()関数を使って行ってください。

## ３．４．３．14　DshFreeTLIMIT_ERR_LIST() - S2F46 応答情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_ERR_LIST(
      TLIMIT_ERR_LIST *pinfo            // ﾒﾓﾘを開放したい変数ﾘﾐｯﾄ情報ﾘｽﾄ構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_ERR_LIST (
        ByRef pinfo As dsh_info.TLIMIT_ERR_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_ERR_LIST(
        ref TLIMIT_ERR_LIST pinfo );
```

（2）引数

pinfo

メモリを解放したい S2F46 応答情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_ERR_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。

## ３．４．３．15 DshInitTLIMIT_ERR_INFO () - 変数リミット応答情報の初期化

（1）呼出書式

[c,C++]

```
API void APIX   DshInitTLIMIT_ERR_INFO(
        TLIMIT_ERR_INFO *info,              // ｴﾗｰ情報格納構造体のﾎﾟｲﾝﾀ
        TVID        vid,                    // 対象変数 ID
        int         lvack,                  // ackﾃﾞｰﾀ
        int         lmt_count               // ｴﾗｰ情報の数（個数  0,1,2...）
);
```

[.NET VB]

```
Sub DshInitTLIMIT_ERR_INFO (
          ByRef pinfo As dsh_info.TLIMIT_ERR_INFO,
          ByVal vid As Int32,
          ByVal lvack As Int32,
          ByVal lmt_count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_ERR_INFO(
          ref TLIMIT_ERR_INFO pinfo,
          uint vid,
          int lvack,
          int lmt_count );
```

（2）引数

info

TLIMIT_ERR_INFO 応答情報構造体のポインタです。

vid

対象となった変数 ID です。

lvack

lvack - ACK の値です。

lmt_count

変数リミットエラー情報の数です。 = 0 の場合はエラー情報がないことになります。

（3）戻り値

なし。

（4）説明

本関数は、S2F45 に対する応答メッセージのための応答情報を TLIMIT_ERR_INFO 構造体に初期設定します。
infoで指定された構造体の lvackメンバーに引数 lvackの値を設定し、lmt_count メンバーに引数 lmt_count の値を設定します。
そして、もし、lmt_count > 0 の場合は、limit_id、limitack 格納のために lmt_count 分の領域を設けます。

err_info へのリミットエラー情報（limit_id, limitack）の設定には DshPutTLIMIT_ERR_ID()関数を使用します。

## ３．４．３．16 DshPutTLIMIT_ERR_ID () - 変数リミット応答情報の設定

（1）呼出書式

[c,C++]

```
API void APIX   DshPutTLIMIT_ERR_ID (
        TLIMIT_ERR_INFO  *info,          // ｴﾗｰ情報格納構造体ﾎﾟｲﾝﾀ
        int  limit_id,                   // ﾘﾐｯﾄ結果設定対象ﾘﾐｯﾄ ID です。
        int  limiack                     // 同 ack
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_ID (
        ByRef pinfo As dsh_info.TLIMIT_ERR_INFO,
        ByVal limit_id As Int32,
        ByVal limitack As Int32) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_ERR_ID(
        ref TLIMIT_ERR_INFO pinfo,
        int limit_id,
        int limitack );
```

（2）引数

info

変数リミット応答情報構造体のポインタです。

limit_id

リミットデータに与えられたリミット ID です。

limitack

リミットデータににによるチェック結果によるリミット ACK です。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | ﾘｽﾄが満杯で設定できなかった。 |

（4）説明

本関数は、info 内の limit_info の先頭から limit_id の空きを探します。
もし、空きがなければ、(-1)を返却します。
もし、空きがあれば、その空きのリスト位置に limit_id と limitack を格納し、0 を返却します。本関数の実行前に DshInitTLIMIT_ERR_INFO()関数を使って info を初期化しておく必要があります。

## ３．４．３．17　DshFreeTLIMIT_ERR_INFO() - S2F46 応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_ERR_INFO(
      TLIMIT_ERR_INFO *pinfo            // ﾒﾓﾘを開放したい S2F46 応答情報ﾘｽﾄ構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_ERR_INFO (
        ByRef info As dsh_info.TLIMIT_ERR_INFO)
```

[.NET C#]

```
void DshFreeTLIMIT_ERR_INFO(
        ref TLIMIT_ERR_INFO info);
```

（2）引数

pinfo

メモリを解放したい S2F46 応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。

## ３．４．３．18　DshMakeS2F45Response( ) - S2F45 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS2F45Response(
        TLIMIT_INFO *info,              // 変数ﾘﾐｯﾄ情報が格納されている構造体のﾎﾟｲﾝﾀ
        TLIMIT_ERR_LIST *erinfo,        // S2F46 応答情報が格納されている構造体ﾎﾟｲﾝﾀ
        DSHMSG *smsg,                   // S2F46 ﾒｯｾｰｼﾞを格納するﾒｯｾｰｼﾞ構造体のﾎﾟｲﾝﾀ
        BYTE   *buff,                   // S2F46 のﾃｷｽﾄ格納ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    buff_size                // buff のﾊﾞｲﾄｻｲｽﾞ
);
```

[.NET VB]

```
Function DshMakeS2F45Response (
          ByRef info As dsh_info.TLIMIT_LIST,
          ByRef erlist As dsh_info.TLIMIT_ERR_LIST,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS2F45Response(
          ref TLIMIT_LIST info,
          ref TLIMIT_ERR_LIST erlist,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

info

変数リミット情報が格納されている構造体のポインタです。

erinfo

S2F46 応答情報（VID 毎のﾁｪｯｸ結果）が格納されているリストの構造体のポインタです。

msg

S2F46 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S2F46 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に生成できた。 |
| (-1) | 生成できなかった。(buff 領域不足) |

（4）説明

S2F45 に対する S2F46 応答メッセージを erinfo に含まれる変数リミット応答情報に従って作成します。
応答情報は、変数 ID 別にチェック結果が erinfo のリスト内に格納されていますので、それに従って S2F46、

メッセージを作成します。

## ３．４．３．19　DshDecodeS2F48() - S2F48 を変数リミット属性情報リストにデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS2F48(
        DSHMSG *smsg,                    // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TLIMIT_RSP_LIST *list            // ﾃﾞｺｰﾄﾞした情報ﾘｽﾄを格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS2F48 (
         ByRef smsg As dshdr2.DSHMSG,
         ByRef list As dsh_info.TLIMIT_RSP_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS2F48(
         ref DSHMSG smsg,
         ref TLIMIT_RSP_LIST list );
```

（2）引数

smsg

S2F48 の SECS ﾒｯｾｰｼﾞ情報構造体のポインタです。

list

デコードした結果の変数リミット属性情報を格納する構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にﾃﾞｺｰﾄﾞできた。 |
| (-1) | smsg を正しくﾃﾞｺｰﾄﾞできなかった。 |

（4）説明

S2F48 メッセージに含まれる変数リミット属性情報リストを、ユーザプログラムが処理しやすい TLIMIT_RSP_LIST 構造体の中にデコードします。TLIMIT_RSP_LIST 内に複数の TLIMIT_RSP_INFO 構造体の情報が含まれます。

変数リミット構造体メンバーのために使用するメモリは全て DSHGEM-LIB 側が準備します。

従って、情報処理終了後は、list のメモリを DshFreeTLIMIT_RSP_LIST()関数を使って解放してください。

```
smsg S2F48
  L,m
    L,2
      VID
      L,p
       UNITS,
       LIMITMIN
       LIMITMAX
       L,n
         L,3
           LIMITID1
```

UPPERDB1
LOWERDB1
.
.

## ３．４．３．20　DshFreeTLIMIT_RSP_LIST() - 変数リミット応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_RSP_LIST(
      TLIMIT_RSP_LIST *pinfo            // ﾒﾓﾘを開放したい変数ﾘﾐｯﾄ情報ﾘｽﾄ構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_RSP_LIST (
        ByRef list As dsh_info.TLIMIT_RSP_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_RSP_LIST(
        ref TLIMIT_RSP_LIST list );
```

（2）引数

pinfo

メモリを解放したい変数リミット応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_RSP_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。

## ３．４．３．21　DshFreeTVLIMIT_EVENT_INFO() - リミット監視結果情報構造体メモリ開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTVLIMIT_EVENT_INFO(
      TVLIMIT_EVENT_INFO *pinfo  // ﾒﾓﾘを開放したいﾘﾐｯﾄ結果情報構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTVLIMIT_EVENT_INFO (
        ByRef info As dsh_info.TVLIMIT_EVENT_INFO)
```

[.NET C#]

```
void DshFreeTVLIMIT_EVENT_INFO(
        ref TVLIMIT_EVENT_INFO info );
```

（2）引数

pinfo

メモリを解放したい変数リミット監視結果情報構造体のポインタです。

（3）戻り値

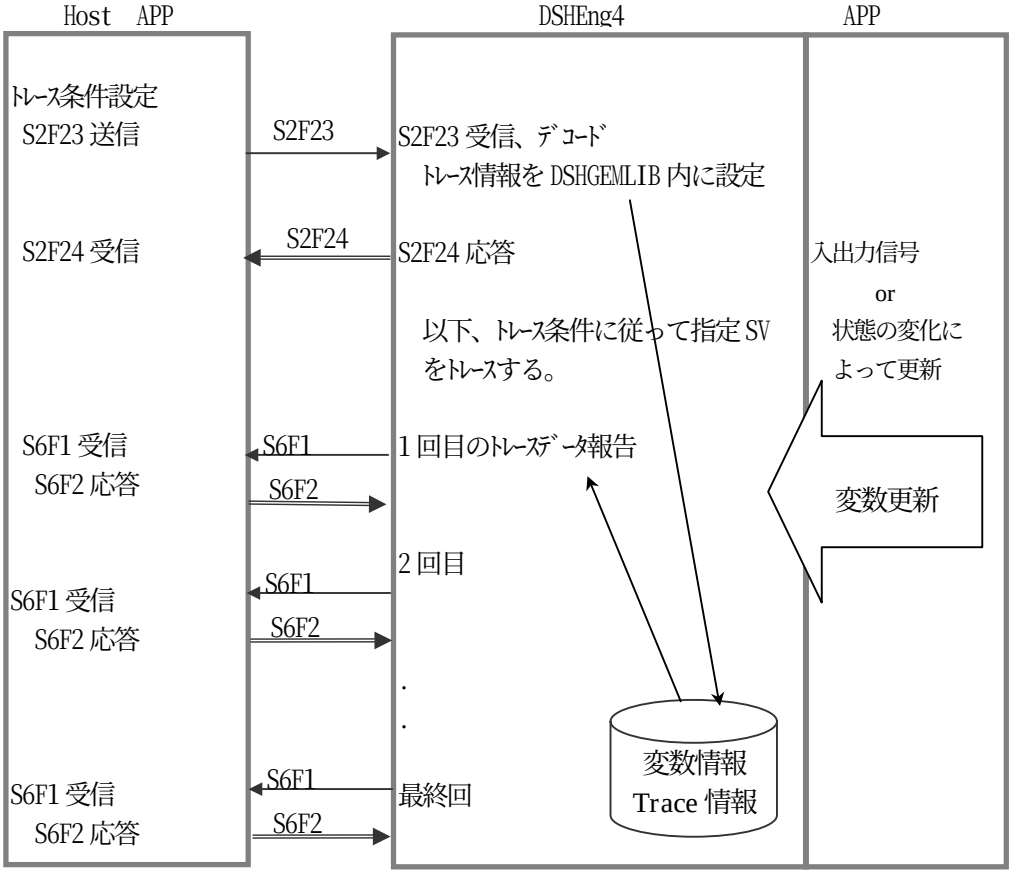
なし。

（4）説明

TVLIMIT_EVENT_INFO 構造体内で情報格納用に使用されているメモリを解放します。

## ３．５．　TR 状態変数トレース情報アクセスサービス関数

トレース情報は、DSHGEM-LIB が管理します。これらの情報をアクセスするために以下の DSHGEM-LIB API 関数を使用します。




（1）情報アクセスと送信 API 関数

トレース情報のアクセスに関連するサービスのための API 関数名は一覧表のとおりです。

| | API 関数名 | 機能 |
|---|---|---|
| 1 | GemAllocTrInfo() | 指定された TRID(ﾄﾚｰｽ ID)のﾄﾚｰｽ情報領域を割当て登録します。 |
| 2 | GemSetTrInfo() | ﾄﾚｰｽ情報を設定・変更します。 |
| 3 | GemGetTrInfo() | TRID 指定でﾄﾚｰｽ情報を取得します。 |
| 4 | GemGetTrInfoX() | TRID 情報ｲﾝﾃﾞｸｽ指定でﾄﾚｰｽ情報を取得します。 |
| 5 | GemDelTrInfo() | TRID 指定でﾄﾚｰｽ情報を削除します。 |
| 6 | GemDelTrInfoX() | TRID 情報ｲﾝﾃﾞｸｽ指定でﾄﾚｰｽ情報を削除します。 |
| 7 | GemGetTrId() | 指定した TRID 情報ｲﾝﾃﾞｸｽの TRID を取得します。 |
| 8 | GemGetTrIndex() | 指定した TRID の情報ｲﾝﾃﾞｸｽを取得します。 |
| 9 | GemEnableTrace() | TRID 指定で TRID の有効状態を指定します。 |
| 10 | GemEnableTraceX() | TRID 情報ｲﾝﾃﾞｸｽ指定で TRID の有効状態を指定します。 |
| 11 | GemSendS2F23() | ﾄﾚｰｽ設定ﾒｯｾｰｼﾞ S2F23 を装置に送信します。 |

TRID インデクスは、DSHGEM-LIB が管理する各 TRID 領域の番号です。このインデクスの値は、GemAllocTrInfo()関数実行時に DSHGEM-LIB によって割当てられ、APP に渡されます。また、TRID 情報の取得時に、情報格納構造体のメンバー、 index に設定されます。

（2）ライブラリ関数

他に APP が使用できるトレース情報処理用関数として、以下の関数があります。

| | API 関数名 | 機能 |
|---|---|---|
| 1 | DshEncodeS2F23() | TTRACE_INFO 構造体のﾄﾚｰｽ情報を S2F23 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。 |
| 2 | DshFreeTTRACE_INFO() | ﾄﾚｰｽ情報が格納されている TTRACE_INFO 構造体の内部で使用されているﾒﾓﾘを開放するための関数。 |
| 3 | DshCopyTTRACE_INFO() | TTRACE_INFO のﾄﾚｰｽ情報を別の構造体ﾒﾓﾘにｺﾋﾟｰします。 |
| 4 | DshDecodeS6F1() | S6F1 ﾒｯｾｰｼﾞに含まれるﾄﾚｰｽ情報を TTRACE_DATA 構造体内にﾃﾞｺｰﾄﾞします。 |
| 5 | DshFreeTTRACE_DATA() | ﾄﾚｰｽﾃﾞｰﾀ情報が含まれる TTRACE_DATA 構造体内に使用されているﾒﾓﾘを開放します。 |
| 6 | DshMakeS6F1Response() | S6F1 に対する S6F2 ﾒｯｾｰｼﾞを作るための関数です。 |
| 7 | DshInitTTRACE_INFO() | TTRACE_INFO 構造体内にﾄﾚｰｽ情報を設定するための初期化関数です。<br>ﾄﾚｰｽ ID，間隔時間などの設定を行います。 |
| 8 | DshPutTTRACE_INFO() | TTRACE_INFO 構造体内の SVID ﾘｽﾄの指定位置に 1 個の SVID を設定します。 |

（3）ユーザ作成ライブラリ関数

| | ﾗｲﾌﾞﾗﾘ関数名 | 機能 |
|---|---|---|
| 1 | DshResponseS2F24() | S2F23 トレース設定応答 |
| 2 | DshResponseS6F2() | S6F1 トレースデータ送信応答 |

### 3．5．1　使用する情報格納構造体

（1）トレース設定情報用

トレース情報を操作する関数は、共通の情報格納のための TTRACE_INFO 構造体を使用します。
TTRACE_INFO とのその内部で使用する他の構造体は下記のとおりです。

```
typedef struct{
        int     index;
        int     state;
        char    *name;                  // trace name
        char    *trid;                  // trace id
        int     format;                 // trace id format
        int     asize;                  // trace id array size
        int     max_asize;
        char    *dsper;                 // trace 時間周期
        int     dsper_time;             // trace 時間周期-数値
        int     totsmp;                 // total sample 数
        int     tot_fmt;
        int     tot_asize;
        int     repgsz;                 // report group size
        int     gsz_fmt;
        int     gsz_asize;
        int     svid_count;             // svid list の size
        TSVID   *svid_list;             // svid list
} TTRACE_INFO;
```

（2）トレースデータ情報

装置からの S6F1 メッセージをデコードしたトレースデータを格納するための構造体です。

```
typedef struct{
        void        *trid;              // trace id
        int         format;             // trace id format
        int         asize;              // trace id array size
        int         smpln;              // sampling no.
        char        *stime;             // start time
        int         count;              // no. of data
        TTRACE_SV   **sv_list;          // sv ptr list
} TTRACE_DATA;

typedef struct{
        int         format;
        int         asize;
        void        *sv;                 // Status Variable data
} TTRACE_SV;
```

## ３．５．２　TRACE トレース情報アクセス関数

### ３．５．２．１　GemAllocTrInfo() - トレース情報の登録

（1）呼出書式

[c,C++]

```
API int APIX GemAllocTrInfo(
        int   eqid,                 // 通信対象装置 ID(0,1,2,...)
        char *trid,                 // ﾄﾚｰｽ ID 格納領域のﾎﾟｲﾝﾀ
        int   *index                // 得られた情報領域ｲﾝﾃﾞｸｽ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemAllocTrInfo (
        ByVal eqid As Int32,
        ByVal trid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemAllocTrInfo(
        int eqid,
        byte[] trid,
        ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

登録したいトレース ID が格納されているポインタです。

index

登録された情報領域のインデクス値が格納される領域のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に登録できた。 |
| 1 | 指定された TRID は既に登録されていた。 |
| (-1) | 登録できなかった。 |

（4）説明

トレース情報を新規にシステムに登録するための関数です。
登録は、引数 trid で与えられるトレース ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデクスが設定返却されます。もし、trid に指定されたトレースが既に登録済みであった場合には関数の戻り値 =1 を返却します。index には既に登録されている情報領域のインデクスが設定されます。

得られたインデクスを使って、情報の設定、取得、削除などのアクセスを行うことができます。

## ３．５．２．２　GemSetTrInfo()　- トレース情報の設定
## .　　GemSetTrInfoX() - インデクス指定によるトレース情報の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetTrInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        TTRACE_INFO *pinfo                  // ﾄﾚｰｽ情報格納構造体のﾎﾟｲﾝﾀ
);

API int APIX GemSetTrInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // GemAllocTrInfo()で得られたｲﾝﾃﾞｸｽ値
        TTRACE_INFO *pinfo                  // ﾄﾚｰｽ情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetTrInfo (
        ByVal eqid As Int32,
        ByRef trinfo As dsh_info.TTRACE_INFO) As Int32

Function GemSetTrInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef trinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int GemSetTrInfo(
        int eqid,
        ref TTRACE_INFO trinfo );

int GemSetTrInfoX(
        int eqid,
        int index,
        ref TTRACE_INFO trinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

pinfo

設定したいトレース情報が格納されている格納構造体領域のポインタです。

index

トレース情報のインデクスです。登録時に GemAllocTrInfo()関数によって与えられます。
インデクスは、トレース ID から GemGetTrdIndex()関数で取得することができます。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |

| | |
|---|---|
| (-1) | 設定できなかった。 |

（4）説明

本関数は、pinfo に格納されているトレース情報の設定・変更に使用します。

引数 pinfo 内の trid メンバーに指定されるトレース ID の情報として設定されます。

pinfo 内には TRID の他、 トレース対象となる変数 ID、トレース周期、合計サンプル数などの情報が含まれます。
指定した TRID が既に登録済みである場合には、pinfo 内の情報に書き換えられます。

pinfo 内の TRID が未登録であった場合は、登録手続きをしてからトレース情報を設定します。
(GemAllocTrInfo()関数で行われる登録と同じ登録が行われます。)

### ３．５．２．３ GemGetTrInfo() - トレース情報の取得
### . GemGetTrInfoX() - インデクス指定でのトレース情報の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetTrInfo(
        int         eqid,                   // 通信対象装置ID(0,1,2,...)
        char        *trid,                  // TRIDが格納されている領域のﾎﾟｲﾝﾀ
        TTRACE_INFO *pinfo                  // ﾄﾚｰｽ情報を格納する構造体のﾎﾟｲﾝﾀ
);

API int APIX GemGetTrInfoX(
        int         eqid,                   // 通信対象装置ID(0,1,2,...)
        int         index,                  // TRID情報のインデクス
        TTRACE_INFO *pinfo                  // ﾄﾚｰｽ情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetTrInfo (
        ByVal eqid As Int32,
        ByVal trid As String,
        ByRef trinfo As dsh_info.TTRACE_INFO) As Int32

Function GemGetTrInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef trinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int GemGetTrInfo(
        int eqid,
        byte[] trid,
        ref TTRACE_INFO trinfo );

int GemGetTrInfoX(
        int eqid,
        int index,
        ref TTRACE_INFO trinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

TRIDが格納されている領域のポインタです。

pinfo

取得したトレース情報を格納する構造体領域のポインタです。

index

TRID情報のインデクスです。登録時にGemAllocTrInfo()関数によって与えられます。

インデクスは、TRID から GemGetTrIdIndex()関数で取得することができます。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に取得できた。 |
| (-1) | 取得できなかった。(TRID が未登録であった。) |

（4）説明

trid に指定されている TRID のトレース情報を取得し、pinfo 構造体に格納します。

TTRACE_INFO 構造体の中に情報を格納するために必要なメモリは、DSHGEM-LIB が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、trid, vid などのためのメモリは DSHGEM-LIB が準備します。
これらのメモリは、使用後、ユーザが DSHGEM-LIB の API 関数を使って次のように開放してください。

```
TTRACE_INFO          *pinfo;

if ( GemGetTrInfo( eqid, tridinfo ) == 0 ){
    pinfo の処理
    処理終了後
    DshFreeTTRACE_INFO( pinfo );          // pinfo 内に使用されているﾒﾓﾘの開放
}
```

## ３．５．２．４　GemDelTrInfo() - トレースの削除
## .　　GemDelTrInfoX() - インデクスでのトレースの削除

（1）呼出書式

[c,C++]

```
API int APIX GemDelTrInfo(
        int       eqid,              // 通信対象装置ID(0,1,2,...)
        char      *trid              // TRIDが格納されている領域のポインタ
);

API int APIX GemDelTrInfo(
        int       eqid,              // 通信対象装置ID(0,1,2,...)
        int       index              // ｲﾝﾃﾞｸｽ(0,1,2....)
);
```

[.NET VB]

```
Function GemDelTrInfo (
        ByVal eqid As Int32,
        ByVal trid As String) As Int32

Function GemDelTrInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32) As Int32
```

[.NET C#]

```
int GemDelTrInfo(
        int eqid,
        byte[] trid );

int GemDelTrInfoX(
        int eqid,
        int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

TRIDが格納されている領域のポインタです。

index

削除したいトレース情報のインデクスです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に削除できた。 |
| (-1) | TRIDが未登録であった。 |

（4）説明

tridまたはindexに指定されているTRIDのトレース情報を登録から削除します。

## ３．５．２．５　GemEnableTrace() - トレース有効状態の設定
## .　　GemEnableTraceX() - インデクスでのトレース有効状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemEnableTrace(
        int         eqid,
        char        *trid                   // TRIDが格納されている領域のポインタ
);

API int APIX GemSEnableTraceX(
        int         eqid,
        int         index                   // TRID情報のインデクス
);
```

[.NET VB]

```
Function GemEnableTrace (
        ByVal eqid As Int32,
        ByVal trid As String) As Int32

Function GemEnableTraceX (
        ByVal eqid As Int32,
        ByVal index As Int32) As Int32
```

[.NET C#]

```
int GemEnableTrace(
        int eqid,
        byte[] trid );

int GemEnableTraceX(
        int eqid,
        int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

TRID（トレースID）が格納されている領域のポインタです。

index

TRID情報のインデクスです。登録時にGemAllocTrInfo()関数によって与えられます。
インデクスは、TRIDからGemGetTrIdIndex()関数で取得することができます。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に有効状態に設定できた。 |
| 1 | 既に有効状態になっていた。 |
| (-1) | TRIDが未登録であった。 |

（4）説明

トレース情報の有効状態を設定します。
トレース状態が有効に設定されると、DSHGEM-LIB は、trid のために登録されているトレース情報に従って、登録されている VID 変数のトレースを開始します。トレースされた情報は、SS6F1 メッセージを使って装置に送信されます。

トレース開始条件としては、トレース情報が登録されており、トレースが休止状態であることになります。

既に有効になっていた場合には、戻り値 = 1 を返却します。

## ３．５．２．６　GemGetTrList() - 全登録トレース ID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetTrList(
        int          eqid,                  // 通信対象装置 ID(0,1,2,...)
        TTEXT_DLIST **list                  // 取得ﾘｽﾄ格納ﾎﾟｲﾝﾀの格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetTrList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int GemGetTrList(
        int eqid,
        IntPtr list );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

取得できた TRID が格納されている TTEXT_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に取得できた。 |
| (-1) | 取得できなかった。 |

（4）説明

システムに登録されている TRID とその名前を TTEXT_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで TRID 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTText_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        char        **id_list;          // 取得できた ID 格納用配列
        char        **name_list;        // 取得できた名前格納ポインタ配列
        }TTEXT_DLIST;
```

## ３．５．２．７　GemGetTrId() - インデクスから TRACEID（トレース ID）の取得

（1）呼出書式

[c,C++]

```
API int  APIX GemGetTrId(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     index,                  // TRACE 情報のインデクス
        char    *trid                   // 取得した TRACEID を格納する領域のポインタ
);
```

[.NET VB]

```
Function GemGetTrId (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal trid As String) As Int32
```

[.NET C#]

```
int GemGetTrId(
        int eqid,
        int index,
        byte[] trid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

index

TRACE 情報のインデクスです。登録時に GemAllocTrInfo()関数によって与えられます。
インデクスは、TRACEID から GemGetTrIdIndex()関数で取得することができます。

trid

TRACEID を格納するための領域のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に取得できた。 |
| (-1) | 取得できなかった。(未登録) |

（4）説明

TRACE 情報のインデクスから TRACEID（トレース ID）を取得し、trid に格納します。
正常に取得できた場合は関数戻り値として 0 が返却されます。

## ３．５．２．８　GemGetTrIdIndex() -TRACEID（トレース ID）からインデクスの取得

（1）呼出書式

[c,C++]

```
API int  APIX GemGetTrIdIndex(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        char    *trid,                  // TRACEID が格納されている領域のポインタ
        int     *index                      // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function GemGetTrIdIndex (
        ByVal eqid As Int32,
        ByVal trid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemGetTrIdIndex(
        int eqid,
        byte[] trid,
        ref int index);
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

インデクスを取得したい対象の TRACEID が格納されている領域のポインタです。

index

取得したインデクスの値を格納するための領域のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に取得できた。 |
| (-1) | 取得できなかった。（未登録） |

（4）説明

trid に指定される TRACEID（トレース ID）から TRACE 情報インデクスを取得するための関数です。
取得されたインデクスは index で指定された領域に格納されます。
正常に取得できた場合は関数戻り値として 0 が返却されます。

## ３．５．２．９　GemSendS2F23() - トレース設定情報メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F23(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TTRACE_INFO *info,                  // ﾄﾚｰｽ設定情報格納構造体のﾎﾟｲﾝﾀ
        INT *tiaack,                        // S2F24 応答 TIAACK 格納用領域ﾎﾟｲﾝﾀ
        int (WINAPI *callback)(),           // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F23 (
        ByVal eqid As Int32,
        ByRef info As dsh_info.TTRACE_INFO,
        ByRef tiaack As Int32,
        ByVal callback As vcallback.callback_S2F23,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F23(
        int eqid,
        ref TTRACE_INFO info,
        ref int tiaack,
        CallbackS2F23 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

info

送信したいトレース設定情報が格納されている構造体のポインタです。
トレース条件と状態変数 ID リストが含まれています。

tiaack

S2F23 に対する応答メッセージ S2F24 の応答 TIAACK を格納する領域のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | (1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。tiaack に応答情報が返却されます。<br>(2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。 |

| | |
|---|---|
| (-1) | 送信できなかった。 |
| (-14) | T3 ﾀｲﾑｱｳﾄを検出した。 |

（4）説明

トレース設定情報を装置に送信するために S2F23 メッセージの送信要求用関数です。
info にトレース設定情報を設定した上で呼出ます。TTRACE_INFO 構造体 info へのトレース情報の設定には以下のライブラリ関数を使用することができます。

DshInitTTRACE_INFO(), DshPutTTRACE_INFO()

tiaack には S2F23 の応答メッセージ S2F24 の応答情報が格納されます。

送信要求から S2F24 応答メッセージ受信までの制御は引数の S2F23 コールバック関数指定の有無によって次のようになります。

| callback の指定 | 制御の流れ |
|---|---|
| なし<br>(=0) | S2F23 送信後、応答ﾒｯｾｰｼﾞ S2F24 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。<br>正常終了であれば、tiaack に応答 TIAACK 情報が渡されます。 |
| あり | 送信要求後、S2F23 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で tiaack に応答 TIAACK 情報が渡されます。<br>ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。 |

（5）コールバック関数

[c,C++]

```
API   int   APIX callback(
        int   eqid,                 // 装置 ID
        int   end_status,           // 実行結果
        int   *tiaack,              // S2F24 応答 TIAACK が格納されているメモリポインタ
        ULONG upara                 // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S2F23(ByVal eqid As Integer, ByVal end_status As Integer, ByRef tiaack As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS2F23(int eqid, int end_status, int* ack, uint upara);
```

end_status には以下の値が設定されます。

| 結果 | 意味 |
|---|---|
| 0 | 正常に送信できた。tiaack 内に S2F24 の応答 TIAACK が格納されます。 |
| (-1) | 送信ｴﾗｰを検出した。 |
| (-14) | T3 ﾀｲﾑｱｳﾄを検出した。 |

## ３．５．３　TR トレース関連ライブラリ関数

### ３．５．３．１　DshDecodeS2F23() - S2F23 をトレース情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS2F23(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TTRACE_INFO *pinfo              // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ格納用
);
```

[.NET VB]

```
Function DshDecodeS2F23 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS2F23(
        ref DSHMSG smsg,
        ref TTRACE_INFO info );
```

（2）引数

smsg

S2F23 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

pinfo

デコードしたトレース情報を格納した構造体ポインタを格納するポインタです。

（3）戻り値

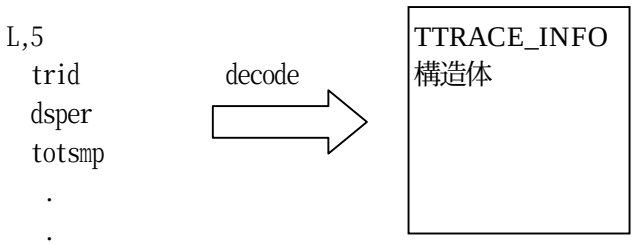
| 戻り値 | 意味 |
|---|---|
| 0 | 正常にﾃﾞｺｰﾄﾞできた。 |
| (-1) | smsg を正しくﾃﾞｺｰﾄﾞできなかった。 |

（4）説明

S2F23 メッセージに含まれるトレース情報を、ユーザプログラムが処理しやすい TTRACE_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTTRACE_INFO()関数を使って開放してください。

smsg S2F23



## ３．５．３．２　DshEncodeS2F23() - トレース情報を S2F23 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS2F23(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S2F23 を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    bufflen,                 // buffer のﾊﾞｲﾄｻｲｽﾞ
        TTRACE_INFO *pinfo              // ｴﾝｺｰﾄﾞしたいﾄﾚｰｽ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS2F23 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef info As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS2F23(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TTRACE_INFO info );
```

（2）引数

smsg

エンコードした S2F23 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S2F23 のテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

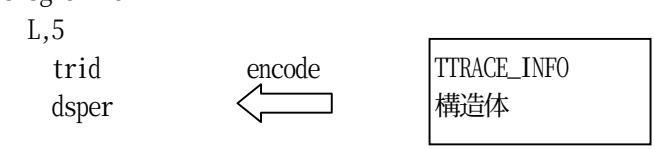
エンコードしたいトレース情報が格納されている構造体のポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にｴﾝｺｰﾄﾞできた。 |
| (-1) | smsg を正しくｴﾝｺｰﾄﾞできなかった。 |

（4）説明

TTRACE_INFO 構造体に格納されているトレース情報を、S2F23 の SECS メッセージにエンコードします。
buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。



totsmp

.

.

### ３．５．３．３　DshFreeTTRACE_INFO() - トレース情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTTRACE_INFO(
      TTRACE_INFO *pinfo          // ﾒﾓﾘを開放したいﾄﾚｰｽ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTTRACE_INFO (
        ByRef info As dsh_info.TTRACE_INFO)
```

[.NET C#]

```
void DshFreeTTRACE_INFO(
        ref TTRACE_INFO info );
```

（2）引数

pinfo

メモリを解放したいトレース情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TTRACE_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TTRACE_INFO の内容を 全て 0 で初期設定します。
pinfo が NULL ならば、何も処理しません。

## ３．５．３．４　DshCopyTTRACE_INFO() - トレース情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
API int APIX DshCopyTTRACE_INFO(
        TTRACE_INFO *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TTRACE_INFO *sinfo                  // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTTRACE_INFO (
          ByRef pinfo As dsh_info.TTRACE_INFO,
          ByRef sinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int DshCopyTTRACE_INFO(
          ref TTRACE_INFO pinfo,
          ref TTRACE_INFO sinfo );
```

（2）引数

pinfo

トレース情報のコピー先構造体メモリのポインタです。

sinfo

コピー元のトレース情報が格納されている構造体メモリのポインタです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常にｺﾋﾟｰできた。 |
| (-1) | sinfo または dinfo の値が NULL だったのでｺﾋﾟｰできなかった。 |

（4）説明

sinfo が指す TTRACE_INFO 構造体内に格納されているトレース情報を dinfo で指定された TTRACE_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用後、DshFreeTTRACE_INFO()関数を使って開放してください。

## ３．５．３．５　DshInitTTRACE_INFO() - トレース S2F43 情報の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTTRACE_INFO(
        TTRACE_INFO *info,                  // ﾄﾚｰｽ情報構造体の格納ﾎﾟｲﾝﾀ
        char    *trid,                      // ﾄﾚｰｽ ID へのﾎﾟｲﾝﾀ
        char    *dsper,                     // ﾃﾞｰﾀ収集時間ﾃﾞｰﾀ格納ﾎﾟｲﾝﾀ(文字列)
        int     totsmp,                     // 総ｻﾝﾌﾟﾙ数
        int     repgsz,                     // ﾚﾎﾟｰﾄｸﾞﾙｰﾌﾟのｻｲｽﾞ
        int     svid_count                  // ﾄﾚｰｽ対象とする SV の数
);
```

[.NET VB]

```
Function DshInitTTRACE_INFO (
        ByRef info As dsh_info.TTRACE_INFO,
        ByVal trid As String,
        ByVal dsper As String,
        ByVal totsmp As Int32,
        ByVal repgsz As Int32,
        ByVal svid_count As Int32) As Int32
```

[.NET C#]

```
int DshInitTTRACE_INFO(
        ref TTRACE_INFO info,
        byte[] trid,
        byte[] dsper,
        int totsmp,
        int repgsz,
        int svid_count );
```

（2）引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

trid

トレース ID（文字列）が格納されている領域のポインタです。

dsper

トレースデータ収集時間を文字列です。“ hhmmsscc ”(8 文字固定)　の文字列で表現します。

hh:時間, mm:分, ss:秒, cc:1/100 秒

totsmp

合計サンプル数を指定します。

repgsz

レポートグループのサイズです。(装置が 1 個の S6F1 で送信するサンプル数)

svid_count

トレース対象の SV の数です。(TTRACE_INFO 内のリストに設定する SV 数)

（3）戻り値

| 戻り値 | 意味 |
|---|---|

| | |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | 引数に指定された値が正しくなかった。 |

（4）説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体の初期設定を行います。
最初に info 内をゼロクリアします。
info のメンバー内にそれぞれの引数を設定します。
TTRACE_INFO メンバー内の以下のメンバーについては固定データを設定します。
format = ICODE_A ( ICODE_A は DSHDR2 ﾄﾞﾗｲﾊﾞｰが定義するﾃﾞｰﾀｱｲﾃﾑの FORMAT の ASCII です)
tot_fmt = ICODE_A, tot_asize=0
gsz_fmt = ICODE_A. gsz_asize=0
引数の値が以下のケースの場合はエラー(-1)を返却します。
trid 文字列長が=0、dsper 文字列長が 8 でない。
totsmp, repgsz または svid_count の値が=0 である。
トレースする SVID の TTRACE_INFO 内への設定は DshPutTTRACE_INFO()関数を使用します。

TSOOL_INFO info 使用が済んだら、情報内に確保して使用したメモリは DshFreeTTRACE_INFO ()関数を使って開放することができます。

## ３．５．３．６　DshPutTTRACE_INFO() - トレース S2F23 情報の SVID 設定

（1）呼出書式

[c,C++]

```
API int APIX DshPutTTRACE_INFO(
        TTRACE_INFO *info,                  // ﾄﾚｰｽ情報構造体の格納ﾎﾟｲﾝﾀ
        int      order,                     // SVID ﾘｽﾄの設定位置(0,1,..)
        TSVID    svid                       // 設定する SVID(状態変数 ID)
);
```

[.NET VB]

```
Function DshPutTTRACE_INFO (
        ByRef info As dsh_info.TTRACE_INFO,
        ByVal order As Int32,
        ByVal svid As Int32) As Int32
```

[.NET C#]

```
int DshPutTTRACE_INFO(
        ref TTRACE_INFO info,
        int order,
        uint svid );
```

（2）引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

order

info 内の svid_list リスト内の設定位置を指定します。位置は先頭が 0 から始まります。

svid

設定したい装置状態変数 ID です。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に設定できた。 |
| (-1) | 設定できなかった( order の値が間違っていた) |

（4）説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体内のメンバー svid_list のリストの order 番目の位置に svid で指定された装置状態変数 ID を設定します。

order の値が、info 内の svid_count の値以上であった場合は、エラー(-1)を返却します。

## ３．５．３．７　DshMakeS2F23Response() - S2F23 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS2F23Response(
        BYTE tiaack,                    // S2F24 に設定する TIAACK です。
        DSHMSG *smsg,                   // S2F24 ﾒｯｾｰｼﾞを格納するﾒｯｾｰｼﾞ構造体のﾎﾟｲﾝﾀ
        BYTE   *buff,                   // S2F24 のﾃｷｽﾄ格納ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    buff_size                // buff のﾊﾞｲﾄｻｲｽﾞ
);
```

[.NET VB]

```
Function DshMakeS2F23Response (
          ByVal tiaack As Int32,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS2F23Response(
          int tiaack,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

tiaack

S2F24 メッセージの TIAACK です。

msg

S2F24 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S2F24 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に生成できた。 |
| (-1) | 生成できなかった。(buff 領域不足) |

（4）説明

S2F23 に対する S2F24 応答メッセージを sinfo に含まれるトレース情報と応答情報に従って作成します。
応答情報は、tiaack を S2F24 の TIAACK として設定します。
tiaack はユーザがトレース情報を評価した結果です。

## ３．５．４　ユーザ作成ライブラリ関数

### ３．５．４．１　DshResponseS2F24() - S2F24 トレース設定応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS2F24(
        int     eqid,                   // 通信対象装置 ID(0,16,...)
        ID_TR   trid,                   // DSHDR2 のﾄﾗﾝｻﾞｸｼｮﾝ ID
        TTRACE_INFO *info,              // ﾄﾚｰｽ設定ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int   tiaack                    // S2F24 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS2F44 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByRef info As dsh_info.TSPOOL_INFO,
          ByRef erinfo As dsh_info.TSPOOL_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS2F24(
          int eqid,
          uint trid,
          ref TTRACE_INFO info,
          int tiaack );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

S2F23 信時に DSHGEMLIB から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

トレース設定モード情報が格納されている構造体のポインタです。

tiaack

送信する応答メッセージ ack の値です。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に送信できた。 |
| (-1) | 送信できなかった。 |

（4）説明

トレース設定メッセージ S2F23 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dsh_ulib.dll)に含まれる関数ですが、ここでは DSHGEMLIB パッケージ

に標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている tiaack 情報から S2F24 メッセージを組み立て、その後、S2F24 メッセージを送信します。

なお、S2F24 メッセージの組み立てに、DshMakeS2F24Response()関数を使用できます。

## ３．５．４．２　DshResponseS6F2() - S6F2 トレースデータ送信応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS6F2(
        int     eqid,                   // 通信対象装置 ID(0,16,...)
        ID_TR   trid,                   // DSHDR2 のﾄﾗﾝｻﾞｸｼｮﾝ ID
        TTRACE_DATA *info,              // ﾄﾚｰｽﾃﾞｰﾀ送信ﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        int     ackc6                   // S6F2 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS6F2 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TTRACE_DATA,
        ByVal ackc6 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS6F2(
        int eqid,
        uint trid,
        ref TTRACE_DATA info,
        int ackc6 );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

S6F1 信時に DSHGEMLIB から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

受信したトレースデータ送信S6F1に含まれていたレポート情報が格納されている構造体のポインタです。

ackc6

送信する応答メッセージ ack の値です。

（3）戻り値

| 戻り値 | 意味 |
|---|---|
| 0 | 正常に送信できた。 |
| (-1) | 送信できなかった。 |

（4）説明

トレースデータ送信メッセージ S6F1 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dsh_ulib.dll)に含まれる関数ですが、ここでは DSHGEMLIB パッケージに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されているackc6情報からS6F2メッセージを組み立て、その後、S6F2メッセージを送信します。